IEICE TRANSACTIONS

on Information and Systems

DOI:10.1587/transinf.2023EDP7201

Publicized:2024/06/11

This advance publication article will be replaced by the finalized version after proofreading.

A PUBLICATION OF THE INFORMATION AND SYSTEMS SOCIETY

The Institute of Electronics, Information and Communication Engineers

Kikai-Shinko-Kaikan Bldg., 5-8, Shibakoen 3 chome, Minato-ku, TOKYO, 105-0011 JAPAN

PAPER

Evaluating Introduction of Systems by Goal Dependency Modeling

Haruhiko KAIYA[†], *Nonmember*, **Shinpei OGATA**[††], *and* **Shinpei HAYASHI**[†††], *Members*

SUMMARY Before introducing systems to an activity in a business or in daily life, the effects of these systems should first be carefully examined by analysts. Thus, methods for examining such effects are required at the early stage of requirements analysis. In this study, we propose and evaluate an analysis method using a modeling notation for this purpose, called goal dependency modeling and analysis (GDMA). In an activity, an actor, such as a person or a system, expects a goal to be achieved. The actor or another actor will achieve this goal. We focus herein on such a goal and the two different roles played by the actors. In GDMA, the dependencies in the roles of the two actors about a goal are mainly represented. GDMA enables analysts to observe the change of actors, their expectations, and abilities by using metrics. Each metric is defined on the basis of the GDMA meta-model. Therefore, GDMA enables them to decide whether the change is good or bad both quantitatively and qualitatively for the people. We evaluate GDMA by describing models of the actual system introduction written in the literatures and explain the effects caused by this introduction. In addition, CASE tools are crucial in efficiently and accurately performing GDMA. Hence, we develop its tools by extending an existing UML modeling tool.
key words: *Goal-oriented requirements engineering, metrics, CASE tools*

1. Introduction

People participate in many activities, such as selling goods for businesses and taking care of elderly people every day. When an artificial element like an information system is introduced into these kinds of activity, the system should be valuable to the humans and the organizations involved in the activity. One of the goals of the early requirements analysis is to clarify whether or not such a system is indeed valuable for the parties involved before developing and introducing it. However, only a few methods can play such a role in requirements engineering techniques. One exception is the iStar modeling notation [1] and its variations [2]. iStar contains two novel concepts, that is, goal dependency and contribution links about quality characteristics. In iStar, a person, an organization or an artificial element (e.g., a system) is called an actor. The concepts help stakeholders understand an activity, the actors, and their dependencies. Although both concepts are important, the concepts make the iStar and its variation models too complicated to identify the improvement caused by introducing systems. This issue is insignificant because the main role of iStar models is to improve the understanding of stakeholders about their activity. Therefore, the concern about many model instances of iStar and its variations not always rigorously following their syntax is not a problem.

We want stakeholders to clearly know whether or not the system introduction is valuable before developing and introducing that system. To satisfy this requirement, we need a new modeling notation that is more formal and simple than iStar and its variations. Thus, in this work, we propose and evaluate a notation, called goal dependency modeling and analysis (GDMA), in which the actors in an activity and these actors' goals are modeled in the same manner as in iStar. Four metrics related to people's gains and losses are calculated on the basis of the model. The metric changes let the stakeholders know whether or not the activity will become better or worse than ever. Introducing new systems is an example of a change in activity. GDMA systematically enables analysts to predict whether or not the introduced systems are valuable. The goals achieved by systems become the bases of their specifications. To calculate the model metrics, its syntax is rigorously defined using a meta-model and several constraints. We develop CASE tools for GDMA to automatically perform a prediction.

As well as the original iStar [1], GDMA and its tools are intended to be used in "early-phase requirements activities" [1]. For example, we expect they will help requirements analysts to consider how the intended systems would meet organizational goals, and to clarify why the systems are needed. Late-phase requirements engineering tasks such as specifying requirements documents precisely, completely and consistently are out of scope in GDMA and its tools. One or more requirements analysts should develop GDMA models and analyze them because stakeholders usually do not have skills to develop models. Example of such stakeholders are customers of systems to be developed and people who participate in a business or life activity supported by the systems. The interaction between the requirements analysts and such stakeholders is of course necessary because the analysts have to know the stakeholders' intention and the activity supported by the system, and to provide analytic results to the stakeholders for getting further information. How to perform such interaction is also out of scope in GDMA.

Although GDMA uses goal dependency, which is an important concept in iStar, the other GDMA concepts are different from those of iStar in some aspects. First, goal delegation is rigorously managed, that is, when a goal exists, it should finally be decomposed into means to achieve the goal. Second, a quality characteristic is not a first-class

[†]The author is a professor in Kanagawa University
[††]The author is an associate professor in Shinshu University
[†††]The author is an associate professor in Tokyo Institute of Technology

object, but a goal attribute.

The contributions of this study are as follows:

1. The ideas of GDMA are proposed.
2. Its meta-model is provided.
3. Its supporting tools are developed and introduced.
4. The tools are explained using the meta-model.
5. Elaborate case studies are presented.

The basic idea of GDMA was informally presented in [3]; however, its meta-model was not defined. Parts of the tools were presented in [4] and [5] as well, but they were not explained on the basis of the meta-model.

The rest of this paper is organized as follows: Section 2 introduces the notation of GDMA and its meta-model and explains the model analysis using metrics, (i.e. CASE tools are crucial in performing modeling and analysis); Section 3 presents the tools used; Section 4 explains the result of the GDMA evaluation; Section 5 discusses the method, the tools and the results of evaluation; Section 6 reviews the related work; and Section 7 summarizes our results.

2. Goal dependency model and its analysis

2.1 Motivating example

As mentioned in the Introduction section, systems introduced into an activity should have a contribution to the people and organizations involved in the activity. GDMA helps us predict whether or not the introduction of systems offers such a contribution. As an example, we use the activity of submitting a paper to a conference to explain GDMA. We regard this activity of more than 30 years ago as an as-is activity and the activity today as a to-be activity. Apparently, the to-be activity is better than the as-is activity with respect to both contributors and organizers.

First, we will describe an as-is situation in the activity. More than 30 years ago, we did not have any ICT-based systems for submitting a paper to a conference and delivering the paper to the conference organizer. When a professor or a researcher wants to submit a paper, he/she must put the paper into print. The printout must then be placed in an envelope and posted. The postal service will deliver the envelope to the conference organizer. This process usually takes approximately 3 days to 1 week from a professor in Asia to an organizer in Europe. The delivering cost is not inexpensive, especially for the express delivery option.

Second, we will describe a to-be situation in the activity. Today, most conferences provide a web-based computer system for submitting and delivering papers. When a professor wants to submit a paper, he/she only needs to prepare a copy of his/her paper in electronic format (e.g., PDF). The file is immediately submitted to the system and delivered to the organizer. In contrast to that of postal services, the running cost of the system is very inexpensive or available for free.

2.2 Goal dependency models in iStar-based notation

Fig. 1 shows an as-is goal dependency model when a professor submits a paper. As mentioned, we had no computer systems for paper submission more than 30 years ago. This model shows this situation. Today, we have many web-based systems for paper submission. The to-be model in Fig. 2 shows today's goal dependencies.

Using these models, we will introduce the syntax and explain the intuitive meaning of the model notation. The notation outline is based on iStar [1]; therefore, we call the notation in these figures as the iStar-based notation. A circle corresponds to an actor (e.g., a person or a system) in an activity. A stereo-type MACHINE is attached when an actor is an artificial element like an information system. The round rectangle between the two actors corresponds to a goal. Each actor and a goal are connected by an arrow. Each actor connected to a goal plays a different role in the goal. The relationship between a goal and two actors shows the delegation of goal achievement. When an actor wants the goal to be achieved, the arrow's direction is from the actor to the goal. In iStar, the actor pursues the role of a "Depender." When an actor will achieve the goal, the arrow's direction is from the goal to the actor. In iStar, this actor plays the role of a "Dependee." In Fig. 1, the goal G1 "A paper submitted" is connected to two actors "Professor" and "Secretary." The "Professor" wants the goal to be achieved, and the "Secretary" will achieve the goal. In our meta-model, the goal between two actors is referred to as "DependGoal."

Although our notation imports the fundamental concept

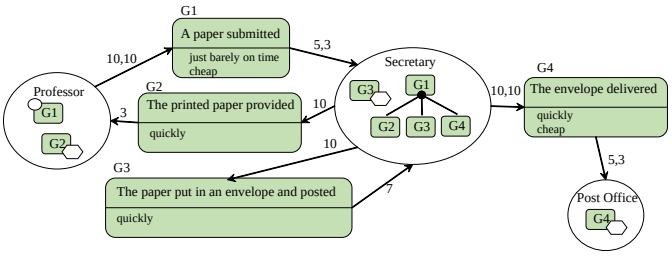


Fig. 1 As-is model of submitting a paper written in the iStar-based notation

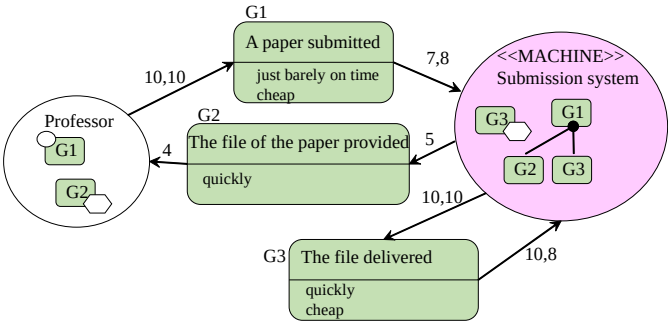


Fig. 2 To-be model of submitting a paper written in the iStar-based notation

of goal dependency from iStar, the other concepts comprising iStar are not used in the same manner. In particular, we do not use four different dependency elements between actors (e.g., goals, tasks, resources and quality.) In addition, our notation has our own concepts; hence, our notation is not an extension of iStar [2].

Our first own concept is the rigorous management of the goal delegation traceability. When an actor wants a DependGoal to be achieved, the actor must contain an OriginGoal or RefinedGoal that corresponds to the DependGoal. The correspondence is represented by the identification number of a goal (e.g., G1, G2, and so on.) An actor sometimes wants a goal to be achieved within an activity without any reasons. This goal is called OriginGoal, in which a small circle is attached to the top-left side. In Figs. 1 and 2, goal G1 "A paper submitted" is an OriginGoal. Although the Professor could have a reason or cause to want to submit a paper outside the activity, these issues are out of the scope of this model. Therefore, G1 is an OriginGoal. In the same manner of iStar, a goal delegated to a "Dependee" can be decomposed into several goals within the actor. The leaf goals of the decomposed hierarchy are called RefinedGoals. Each RefinedGoal must be delegated to one of the actors. In Fig. 1, goal G1 is decomposed into G2, G3, and G4, and each of which is a RefinedGoal. Each RefinedGoal is delegated to the "Professor," "Secretary" and "Post Office." As shown in this example, a RefinedGoal in an actor can be delegated to the actor him/herself.

When an actor will achieve a DependGoal, the actor must contain a MeansGoal or a RefiningGoal that corresponds to the DependGoal. Nothing will be achieved forever if all actors continue the delegation of achieving goals. Therefore, some actors must prepare means to achieve a goal and stop the delegation chain. A MeansGoal in an actor shows that the actor has the means to achieve a goal. In this case, the goal does not have to be delegated to another. The goal of the course does not have to be decomposed into other goals as well. We do not specify the details of the means in our modeling. We only clarify that an actor has some means. In Fig. 1, the "Secretary" has the means to achieve goal G3 stating "The paper put in an envelope and posted." Perhaps, he/she will do them him/herself. As shown in this example, a hexagon icon is put at the right-bottom part of the MeansGoal. If a goal is delegated to an actor, but the actor does not have any means to achieve this goal, the actor may decompose the goal into several goals such that each goal can be delegated to an actor. In this case, a decomposed goal is called a RefiningGoal. And-decomposition and or-decomposition can be used in this goal decomposition. G1 in "Secretary" in Fig. 1 and G1 in "Submission system" in Fig. 2 are examples of RefiningGoals. Only and-decomposition is used in both cases.

Our second own concept touches on the quality characteristics and their quantification. Adverbs expressing quality characteristics like "efficiently" and "accurately" are used to qualify the verbs. Although a goal is not a verb, most goals are represented by using nouns and a verb (e.g., "A paper submitted"). An adverb is used to specify the degree of goal achievement in a viewpoint, such as "efficiency" or "accuracy," and each goal generally has several different viewpoints. The meaning of the adverb is unclear when it is not related to a verb. Therefore, we made an adverb that expresses a quality characteristic as a goal attribute in our notation. Figs. 1 and 2 depict a rectangle, which represents a goal, being vertically divided into two parts. In the upper part, a goal is specified by using nouns and a verb in the the past perfect tense form. In the lower part, the quality characteristics are listed line by line. The two quality characteristics of "just barely on time" and "cheap" qualify goal G1 "A paper submitted." The other goals in the figures only have one quality characteristic. The values attached to the arrows incoming and outgoing a DependGoal correspond to the quality characteristics attached to the DependGoal. Each value takes from 1 to 10 integer values. Each value attached to an arrow incoming a DependGoal shows the level of how well an actor wants the DependGoal with respect to the corresponding quality characteristic. In this paper, the value is sometimes referred to as the want-level. The actor really or supremely wants the goal when the value is 10. Each value attached to an arrow outgoing a DependGoal shows the level of how well an actor can achieve the DependGoal with respect to the corresponding quality characteristic. The value is sometimes referred to herein as the can-level. In Fig. 1, the values around DependGoal G4 "The envelop delivered" represent the following extents: "a Secretary" wants G4 to be achieved quickly and inexpensively. However, the "Post Office" achieves G4 not so quickly and not so late and expensive.

As shown in the Secretary in the as-is model, G1 is decomposed into G2, G3 and G4. All these decomposed goals have a quality characteristic "quickly". Because Secretary is not a system but human and actors who achieve the decomposed goals are also human or an organization, the achievement of such goals and the composition of the goals are inherently slow. Especially, the achievement of G3 and G4 is quite slow because it requires physical tasks such as sending the envelope by air mail. The Secretary thus sets the want-level of quickly in the decomposed goals to 10 so that G1 can be achieved as just barely on time as possible. On the other hand of the to-be model in Fig. 2, Submission system decomposes G1 into G2 and G3, and G3 is achieved by itself. Because Submission system is not human but an information system, the achievement of G3 is quick and the composition of G1 is also quick. As a result, G2 may be achieved a little bit slow by Professor although G1 has to be achieved as just barely on time as possible. Therefore, the system sets the want-level of G2 to 5 in the to-be model.

2.3 Meta-model and its instance

The previous subsection intuitively introduced our goal dependency models. We then defined its formal syntax by using a meta-model in Fig. 3. The meta-model was written in a UML-class diagram; thus, a concrete model corresponding

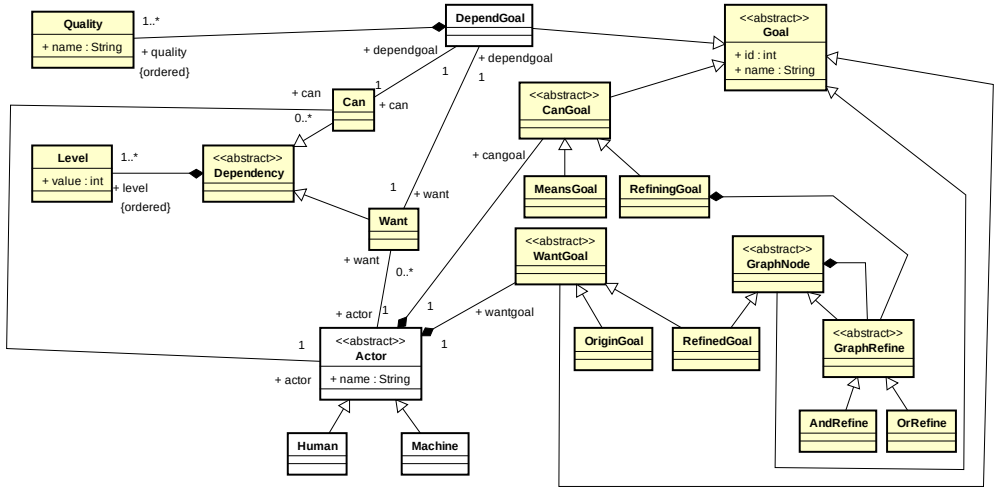


Fig. 3 Meta-model of the goal dependency modeling notation written in the UML-Class Diagram

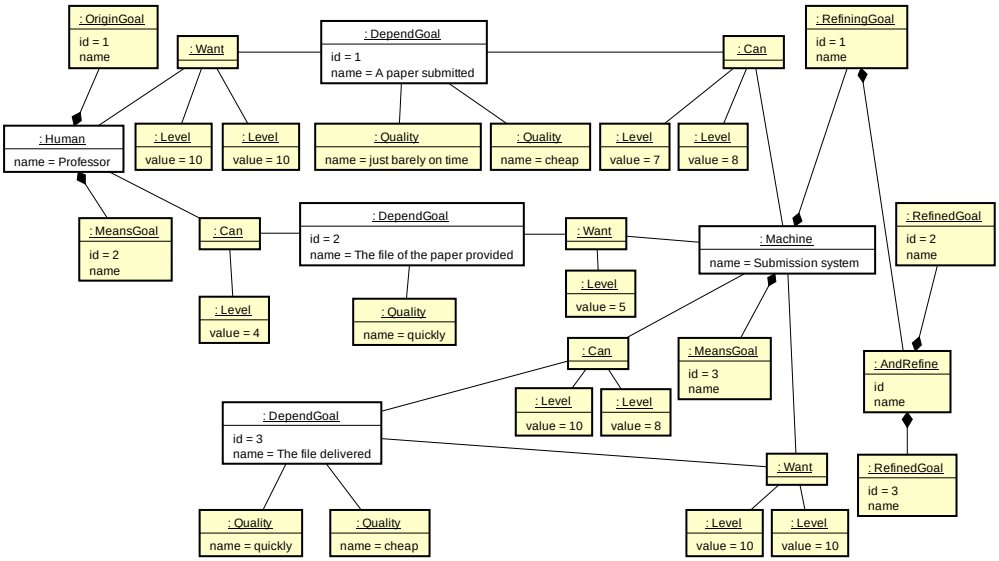


Fig. 4 Model corresponding to the to-be model in Fig. 2 written in the meta-model-based notation (i.e. an instance of the meta-model in Fig. 3)

to Fig. 1 or 2 is represented in an object diagram (Fig. 4).

We will briefly explain this meta-model here. The main elements in the meta-model are “Actor” and its subclasses and “DependGoal” as represented by the white color in Fig. 3. An Actor is either “Human” or “Machine.” This distinction is introduced to clarify the effects of artificial elements introducing an activity. “Machine” corresponds to an artificial element, such as an information system or a hardware element, while “Human” corresponds to an “Actor” which is not a “Machine”. Therefore, a company or an organization is a “Human”. The most important structure in our notation is the relationship between two “Actors” and a “DependGoal.” This relationship is represented by “Actor”, “DependGoal”, “Can”, “Want,” and the associations among them in the meta-model. Each “DependGoal” has more than one quality characteristics representing in “Quality” classes in the meta-model. The “Want” and “Can” classes correspond to the arrows incoming and outgoing a “DependGoal”. Both “Want” and “Can” have to possess several values of a quality level representing in a class “Level”. As shown in the meta-model, there is no direct mapping between “Quality” and “Level”. However, there is an indirect mapping between them. There is an association between “Quality” and “DependGoal”, and the “Quality” is ordered. Therefore, a list of “Quality” instances is like an array of programming languages, and a “DependGoal” instance owns the array. There is one-to-one mapping between “Can” and “DepandGoal”. There is also one-to-one mapping between “Want” and “DependGoal”. In addition, both “Can” and “Want” are subclasses of “Dependency”. In the same way as

"DependGoal", there is an association between "Level" and "Dependency", and the "Level" is ordered. Therefore, a list of "Level" instances is also like an array, and a subclass instance of "Dependency" owns the array. As a result, we can identify an indirect mapping between "Quality" and "Level" by using the indices of such arrays. Note that the lengths of the arrays have to be the same. This constraint is enforced by the Eq.(2) below.

In addition to the meta-model, we need several constraints to specify the valid models. We define these constraints in OCL expressions in Eqs. (1)-(4). Note that $\rightarrow$ in these equations is not the logical implication, but the accessor in the OCL expressions.

$$\begin{aligned} &context\ Level \\ &inv : 1 \leq self.value\ and\ self.value \leq 10 \end{aligned} \tag{1}$$

$$\begin{aligned} &context\ Can \\ &inv : level \rightarrow size() = dependgoal.quality \rightarrow size() \\ &context\ Want \\ &inv : level \rightarrow size() = dependgoal.quality \rightarrow size() \end{aligned} \tag{2}$$

$$\begin{aligned} &context\ Actor \\ &inv : wantgoal \rightarrow forAll(x, y \mid x <> y\ implies\ x.id <> y.id) \\ &inv : want \rightarrow forAll(x, y \mid x <> y\ implies \\ &\qquad x.dependgoal.id <> y.dependgoal.id) \\ &context\ Want \\ &inv : actor.wantgoal \rightarrow exists(c \mid c.id = dependgoal.id) \end{aligned} \tag{3}$$

$$\begin{aligned} &context\ Actor \\ &inv : cangoal \rightarrow forAll(x, y \mid x <> y\ implies\ x.id <> y.id) \\ &inv : can \rightarrow forAll(x, y \mid x <> y\ implies \\ &\qquad x.dependgoal.id <> y.dependgoal.id) \\ &context\ Can \\ &inv : actor.cangoal \rightarrow exists(c \mid c.id = dependgoal.id) \end{aligned} \tag{4}$$

In the "Level", each value should take from 1 to 10 [Eq. (1)]. The number of "Levels" associated to "Can" or "Want" should be the same as the number of "Quality". This constraint is represented in Eq. (2). Each "Actor" can contain several "OriginGoals" and/or "RefinedGoals" generalized into "WantGoals." Each "WantGoal" has to be delegated to an "Actor" via a "DependGoal." This constraint is represented in Eq. (3). In the same manner, each "Actor" can contain several "MeansGoals" and/or "RefiningGoals" generalized into "CanGoals." Each "CanGoal" must be delegated from an "Actor" via a "DependGoal." This constraint is represented in Eq. (4).

The instance of the meta-model in Fig. 4 corresponds to the model in Fig. 2. We call the representation in Fig. 4 as the meta-model-based notation. The model conforms with the meta-model in Fig. 3 and the abovementioned constraints. The instance in Fig. 4 shows that only "DependGoal" instances have both "name" and "id" attributes. Therefore, an "id" attribute becomes a key to access the corresponding "name" attribute. Thus, the "name" attribute of an "OriginalGoal" or "MeansGoal" is empty. The instances of "RefinedGoal" and "RefiningGoal" are intermediate nodes only, and they are not referred from others; hence, their "name" attributes are also empty. Considering the constraint of our prototype tool in Section 3.3, their "name" attributes are not empty in our prototype tool, but instead are filled with a dummy name, such as "Class1" or "Class2." And-decomposition and or-decomposition are represented in the instances of "AndRefine" or "OrRefine", each of which is a subclass of "Goal". We want to use a composite pattern to represent the hierarchical structure. In this figure, the mapping between "Quality" and "Level" instances seems to be expressed only by the placement of objects. However, the mapping is indirectly specified by the orders of "Quality" instances and "Level" instances as mentioned at the end of second paragraph in this subsection.

2.4 Evaluation by using metrics

When the actors and their contributions to an activity are changed, the activity can be better or worse than ever. We evaluate whether or not a snapshot of an activity is better than another by using the models representing each snapshot during the changes. We set the following criteria for the evaluation:

1. When the goals of a human are achieved, and the number of goals is more than ever, we regard this activity to be better than ever.
2. When the goals of a human are achieved, and the quality of achieving these goals is better than ever, we regard this activity to be better than ever.
3. When a human should achieve goals, and the number of goals is more than ever, we regard this activity as worse than ever.
4. When a human should achieve goals, and the quality of achieving these goals is required to be better than ever, we regard this activity to be worse than ever.

The criteria are constructed following two different viewpoints. The first viewpoint is about the quantity of goal achievement and the quality. The second viewpoint is about who wants to achieve a goal and who achieves it. The first and second criteria are obvious, while the other bear some argument. The last two criteria are based on the position that people do not want to work and to be depended on by others. We believe that this position is appropriate at least in a business activity.

We define the four metrics of Average Number of Wants (ANW), Average Gain of Wants (AGW), Average Number of Cans (ANC), and Average Gain of Cans (AGC). Because the metrics correspond to the abovementioned criteria, ANW and ANC show the quantity of goals, and AGW and AGC show the quality. Also, ANW and AGW show the evaluation by dependers, i.e. actors who want to achieve goals, and ANC and AGC show the evaluation by dependees, i.e. actors

who will achieve goals. ANW and ANC are independent of AGW and AGC because the quantity is independent of the quality. All depenters' goals are achieved by a dependee. Therefore, ANW completely depends on ANC if all actors are human. However, non-human actors such as information systems exist in a model because we describe models so that we can examine their introduction. In addition, ANW does not take goals wanted by non-human actors into account. ANC also does not take goals achieved by non-human actors into account. As a result, ANW does not completely depend on ANC, and vice versa. The relationship between AGW and AGC is the same.

Algorithms 1 to 4 in the Appendix are the formal algorithms used to calculate the metrics. We will show their informal definitions and their rationale.

1. ANW: We simply count the number of instances of an "OriginGoal" of a "Human" in average. We do not consider instances of a "RefinedGoal" because they are only subcontract goals derived from a goal assigned by another. We are not so happy when such goals are achieved. We are rather happy when someone else does not assign us to his/her goal.
2. AGW: As exemplified in Fig. 4, a "DependGoal" instance is associated to an instance of "Can" and an instance of "Want." The level of "Want" corresponds to the degree of expectation by an actor who want to achieve the "DependGoal." The level of "Can" corresponds to the degree of ability of an actor who will achieve the "DependGoal." Thus, we focus on the value of the ability divided by the expectation. In the AGW, we only focus on the instances of "DependGoal" wanted by human. The average of these values about the instances is the AGW.
3. ANC: We simply count the number of "MeansGoal" instances of human in average. We regard the means to achieve a goal to be the main effort.
4. AGC: In the AGC, we only focus on the instances of the "DependGoal" achieved by human and on the value of the expectation of an instance divided by the ability to achieve the instance. The average of these values is the AGC.

Table 1 presents the results of metrics about the paper submission activity in Figs. 1 and 2. We briefly explain how to calculate the metrics in the table. The ANC's numerator of the as-is model in Fig. 1 is 1+1+1, i.e. 3. In Fig. 1, Professor, Secretary and Post Office contain goals with a hexagon mark; G2, G3 and G4 respectively. The value 3 corresponds to the number of such goals. Such a hexagon mark indicates that the actor containing the goal with the mark has the means to achieve the goal. The denominator of the ANC corresponds to the number of such actors, that are not a machine or a system. Algorithm 3 in appendix formally represent how to calculate the ANC. Because the algorithm only focuses on the Human actors in the model, the ANC of to-be model in Fig. 2 is $1/1$ as shown in the table. The denominator of the AGC for the as-is model in Fig. 1 is $1 + (2 + 1) + 2$. This value corresponds to the number of quality characteristics in G2, G1, G3 and G4 respectively. Professor will achieve G2, and G2 has one quality characteristic. Secretary will achieve G1 and G3, and G1 and G3 have two and one characteristics respectively. Post Office will achieve G4, and G4 has two characteristics. These numbers of the quality characteristics are summed up, and the result is put at the denominator. The line 8 of algorithm 4 in Appendix corresponds to this summing up. The AGW is also calculated in the similar way as shown in algorithm 2 in Appendix.

All metrics except ANC proved that the to-be model is better than the as-is model because the ANW and the AGW increase, and the AGC decreases. The increase of the ANW means that the goals of the people in this activity are achieved more than ever. The increase of the AGW means that the goals are achieved better than ever. The ANC shows that people in a to-be model have to achieve the same number of goals as people in an as-is model do. The decrease of the AGC means that the people may achieve the goals more comfortable than ever.

Table 1 Metrics in Figs. 1 and 2

	As-is model in Fig. 1	To-be model in Fig. 2
ANW: Average Number of Wants	$\frac{1}{3} = 0.333$	$\frac{1}{1} = 1.000$
AGW: Average Gain of Wants	$\frac{\frac{5}{10}+\frac{3}{10}+\frac{3}{10}+\frac{7}{10}+\frac{5}{10}+\frac{3}{10}}{6} = 0.433$	$\frac{\frac{7}{10}+\frac{8}{10}}{2} = 0.750$
ANC: Average Number of Cans	$\frac{1+1+1}{3} = 1.000$	$\frac{1}{1} = 1.000$
AGC: Average Gain of Cans	$\frac{\frac{10}{5}+\frac{10}{3}+\frac{10}{3}+\frac{10}{7}+\frac{10}{5}+\frac{10}{3}}{1+(2+1)+2} = 2.542$	$\frac{\frac{5}{4}}{1} = 1.250$

3. Modeling tools

As mentioned, CASE tools are crucial in performing an analysis of the system introduction using GDMA. Accordingly, we developed prototypes of these CASE tools to evaluate our idea.

3.1 Class diagram-based notation

When describing a model, the iStar-based notation in Figs. 1 and 2 can easily be used by iStar users; however, not all developers are familiar with iStar. In addition, we have to develop and introduce a specific tool to edit the notation. The meta-model-based notation, such as the instance of the meta-model used in Fig. 4, is precise and can be described using the usual UML modeling tools. However, describing the models in the notation is complicated. We opted for a goal dependency model using the UML-class diagram because instance diagrams, such as those in Fig. 4, are less familiar to many developers than class diagrams. At least in classrooms, tool users could operate them smoothly because

Table 2 Comparison of the meta-model, iStar-based notation, and class diagram-based notation

Meta-model	iStar based-notation	Class diagram-based notation
Human	Circle	Subsystem
Machine	Circle with MACHINE	Subsystem with MACHINE
Goal	Round rectangle with ID number e.g., G1	Class with Gi
DependGoal	Goal with its name	Class with its name
Can, Want	Arrow	Association with one navigability
Level	Arrow's annotation	Constraint of an association
MeansGoal	Small goal in an Actor with hexagon	Class in a subsystem with MEANS
RefiningGoal	Small goal in an Actor	Class in a subsystem
OriginGoal	Small goal in an Actor with circle	Class in a subsystem with ORIGIN
RefinedGoal	Small goal in an Actor	Class in a subsystem
AndRefile	branching	Composition
OrRefile	branching with "or"	Aggregation
Quality	Phrase in a DependGoal	Class with its name and QUALITY

The words in capital letters (e.g., MACHINE) are stereotypes. Gi is also a stereotype (e.g., G1 and G2).

they had already used an UML editor, extended to the tools in other lectures and exercises [4], [5]. In the UML editor, the powerful APIs enable us to extend functionalities during editing class diagrams. Hence, we efficiently implement the modeling tools for GDMA.

Fig. 5 illustrates a model in the class diagram-based notation corresponding to the models in Figs. 2 and 4. Table 2 shows us the correspondence between an element in the iStar-based notation and that in the class diagram-based notation. The ID of a goal is very important; hence, it is represented in a stereotype (e.g., G1 and G2) for emphasis. The quality characteristics are represented in classes because they become goal attributes (Fig. 5).

3.2 Functions of the tools

Our tools provide four kinds of functions.

1. Editor: On the basis of the class diagram-based notation, we can describe the models using the usual UML modeling editors. However, it is not easy for a developer to follow the meta-models and constraints in Section 2.3; hence, we added functions to an existing modeling editor to check the syntax in the meta-model. We also added functions to insert specific stereotypes in Table 2. Fig. 6 shows a snapshot of the editor. In a manner similar to that of usual editors, we can add model elements on the pallet at the top of the editor. The specific stereotypes can be added via another pallet at the bottom of the editor.
2. Metrics calculator: The metrics in Section 2.4 can be automatically calculated as exemplified in Fig. 7. The metrics are automatically updated when a model is modified, and its syntax is correct. A "No Change" annotation is placed even if a metric is not changed according to the editting (Fig. 7); otherwise, "better" or "worse" is placed according the criteria presented in Section 2.4.
3. Delegation tracer: A model mainly contains the trees of goal delegation. The root goal of each tree is an "OriginGoal," and the leaf goals are the "MeansGoals." This tool shows such a tree forward and backward, even from an intermediate goal. Fig. 10 shows a relevant example. In this example, the delegation tree from goal G3 "Soap supplied" is traced forward. Using colors, our tool tells us that goal G3 is finally achieved by the

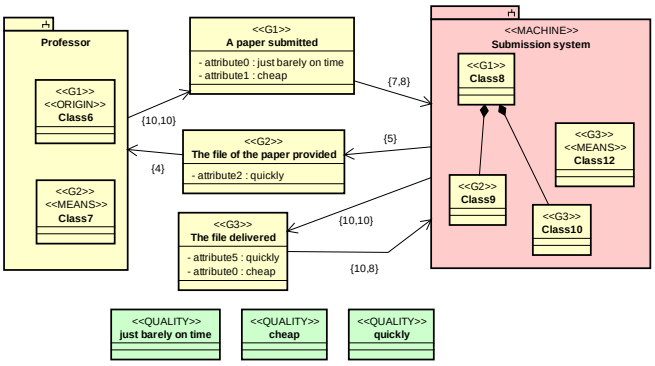


Fig. 5 Model corresponding to the to-be model in Fig. 2 written in the class diagram-based notation

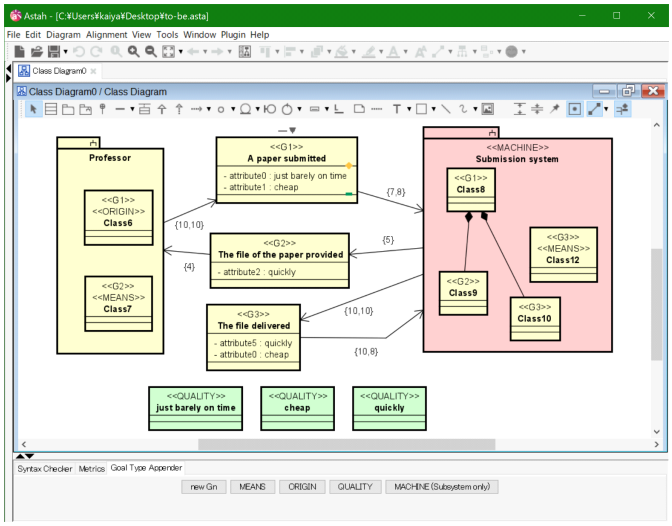


Fig. 6 Snapshot of Editor

"MeansGoal" G1 in "Warehouse" and the "MeansGoal" G4 in "Sensor." This function mitigates the sixth issue in Section 5.

4. Quality tracer: Each goal has several quality attributes, which are usually inherited during the goal refinement specified in "GraphNode" in the meta-model. These attributes are not always inherited because of the rationale of an actor refining the goal. This tool asks us whether or not quality attributes should be inherited when they are not. Fig. 8 shows a relevant example, in which goal G1 has two attributes (i.e., "just barely on time" and "cheap"), while its refined goal G2 only has one (i.e., "quickly"). The tool asks the user whether or not he/she accepts this refinement. When he/she accepts it, the tool remembers it and never asks it. This function mitigates the second issue in Section 5.

3.3 Implementation of the prototype tools

We implemented the tools by extending an existing UML modeling tool, called astah †. astah is not free of charge, but it largely inexpensive, especially for academic users. It costs approximately 400 EUR per year for all the students and staffs in a department. On the basis of the plugin mechanism of astah, the additional functions mentioned earlier can be easily added. Each plugin can be developed using Java because the plugin mechanism is based on OSGi. Note that the tools are manually implemented according to the requirements and design. Model transformation techniques have not been applied to the meta-model in Fig. 3, which is part of the design. Considering the constraint of astah, all classes must have their own names. This is the reason why the goals in the actors have dummy names (e.g., "Class1" and "Class2"), as exemplified in Figs. 8 and 10. These dummy names are automatically placed by our tools.

†https://astah.net/

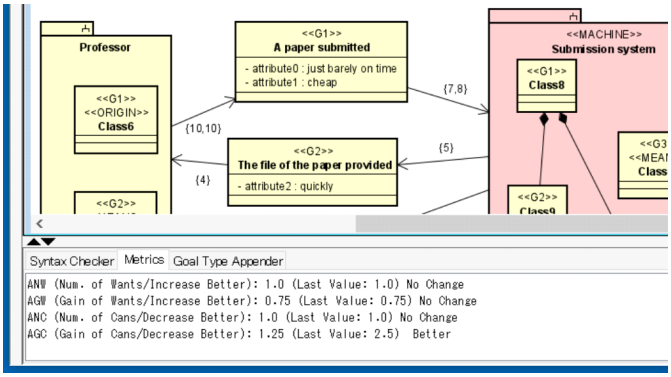


Fig. 7 Metrics calculator

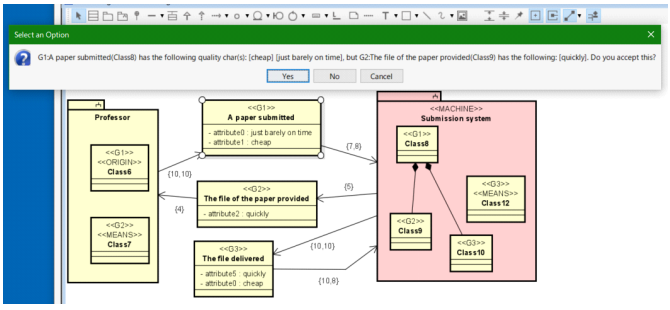


Fig. 8 Quality tracer

4. Case studies

We validate GDMA by analyzing two different cases of system introduction reported in the literatures. The first case is about stock management using sensors in 2016, which was reported as successful in the literature [6]. The second one is about the automated dispatch of ambulances in 1992. This case is widely famous because of the terrible results [7]. Our modeling and analysis can conclude the same results reported in the literatures.

4.1 Good system introduction

Table 3 Metrics of Hagleitner's activity and its evaluation

Metrics	As-is	To-be	To-be is · · ·
ANW	1.00	1.00	The same
AGW	0.56	0.88	Better than as-is
ANC	2.50	2.00	Better than as-is
AGC	1.83	1.23	Better than as-is

A book [6] introduced the case of the stock management in Hagleitner, an Austrian manufacturer of sanitary products and their dispensers. One of Hagleitner's business is supplying liquid soap and paper towels to customers, including those in the healthcare and catering industries. The demand for their services on this aspect increased, making the company fail to supply the necessary soap and towels as soon as possible. Cleaning staffs manually checked the dispensers and reported the lack of soap, etc. The company also needed sufficient stocks in a warehouse because predicting the delivery amount was not easy. The company decided to introduce sensors to monitor the dispensers at the customers' sites. This system introduction enabled Hagleitner to supply the soap and other products on time and minimize the amount of stocks. We modeled the changes they underwent in Figs. 9 and 10.

Fig. 9 shows an as-is model of this activity. Fig. 10 depicts its to-be model. With the help of sensors, the company is now able to immediately identify shortage of soap and other products. As a result, the company did not have to secure a large stock. Table 3 shows the results of the metrics and their evaluations. No new goals of the company and the customers were achieved in the to-be model, but the quality of the existing goals improved. In this business, both the company and the customers must achieve several goals. The quality and the quantity of such goals decreased

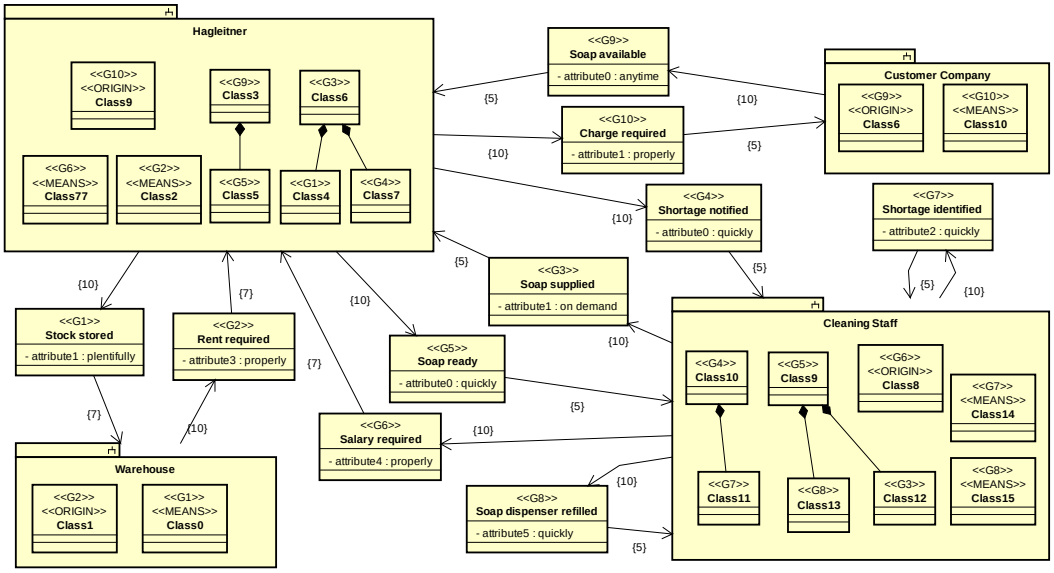


Fig. 9 As-is model of the stock management in Hagleitner

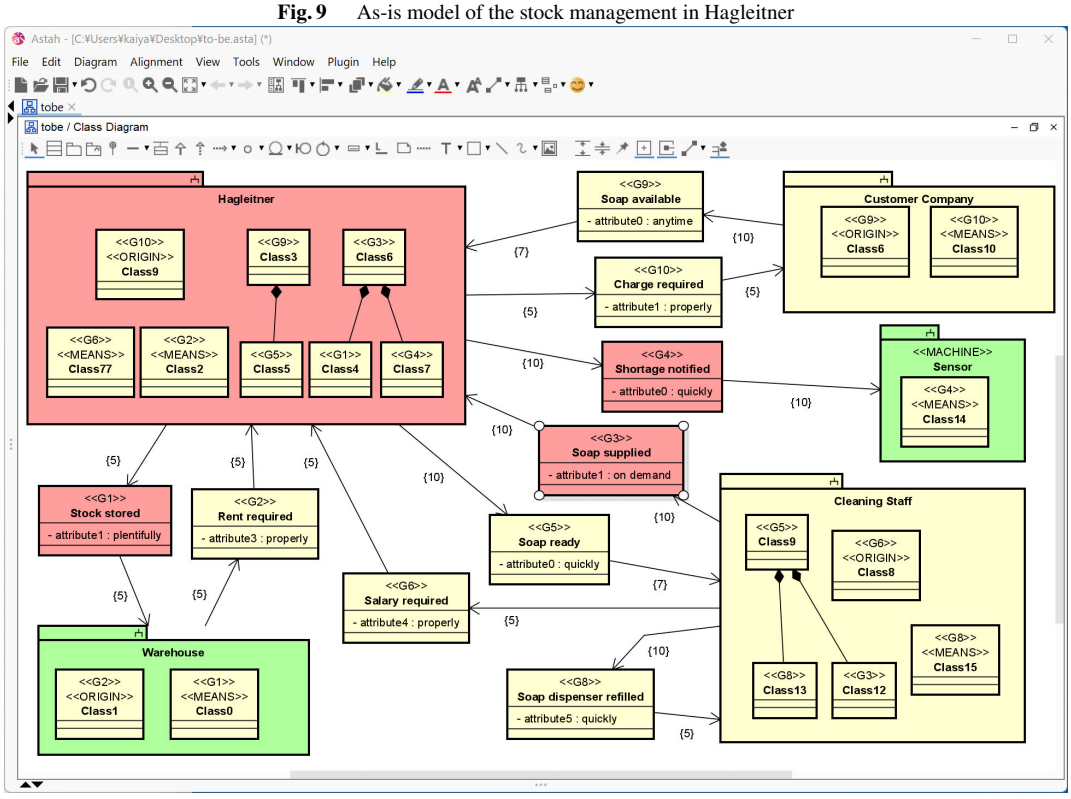


Fig. 10 To-be model of the stock management in Hagleitner, with the result of delegation tracer

Table 4 Metrics of the ambulances dispatch

Metrics	As-is	To-be	To-be is · · ·
ANW	0.14	0.14	The same
AGW	0.79	0.72	Worse than as-is
ANC	2.43	2.00	Better than as-is
AGC	1.31	1.48	Worse than as-is

in the to-be model, that is, the company and the customers may work less than ever. The metrics can reveal these issues. A "MACHINE" by the form of the "Sensor" was introduced. Consequently, the obligation of the "HUMAN" quantitatively and qualitatively decreased. The ANC and the AGC reflected such results. In the book [6], the authors emphasized that business "transparency" is important, and we agree with it. The goal dependency model and the model traceability are a means of showing transparency.

4.2 Bad system introduction

The second case is about the system introduction to the ambulance dispatch activity [7]. This case is called the LAS because it was a project in the London Ambulance Service. A detailed report can be obtained from the site [†]. This introduction caused terrible failures because of the many problems related to technology and organization. We mainly focus herein on the technical problems.

Fig. 11 depicts the as-is model of LAS, while Fig. 12 presents its to-be model. Table 4 shows the results of the metrics and their evaluation. Two systems were introduced in this case, namely AVLS and MDT. AVLS must manage the status of ambulances such that a staff can dispatch an ambulance to an emergency call. In the as-is situation, these tasks were manually achieved by staff using papers and phone calls. The MDT must efficiently send the status of an ambulance via a computer terminal by ambulance crews. Phone calls were used for this task in the as-is situation via the radio operator. These would work well if the technologies were matured, and the crews were sufficiently trained. However, the technologies in 1992 were poor, and the crews were not adequately trained, which were the main reasons for the terrible results of LAS. A reason is modeled as a goal dependency about G15 between AVLS and MDT in Fig. 12. Before introducing the technologies, this goal dependency was established between the radio operator and the ambulance crew (Fig. 11). The difference of the can-levels in G15 between the two models revealed that G15 was achieved worse than ever. Basically, this system introduction is the replacement of the as-is activity; therefore, no new goals were achieved in the to-be model. The ANW in Table 4 reflects this fact. The systems would achieve several goals instead of human in the to-be model. Hence, the obligation of the goal achievement by the human quantitatively decreases, and the ANC reflects this fact. We assumed that this quantitative analysis of the ANC caused a large pitfall. The AGW (i.e., a result of the qualitative analysis) revealed that the system introduction was bad with respect to the quality of the goals of the human. The AGC also proved the same because the introduction asks the human to achieve goals better than ever. In summary, our goal dependency models and their analysis using metrics can explain the failure of this system introduction.

[†]`http://ifs.host.cs.st-andrews.ac.uk/Resources/CaseStudies/LondonAmbulance/LAS-failure-report.pdf` Last accessed Aug. 2021.

5. Discussion, limitation and threats to validity

We will discuss ten issues related to GDMA. The last and eighth issues are related to the evaluation in the previous section, and others are related to GDMA and its tools.

The first issue is how to decide on the value of levels about the quality characteristics. Basically, an analyst must subjectively decide on the values. He/she may start to decide on the can-levels about the "MeansGoal" because these can be predicted on the basis of the ability of the actual means. He/she then decides on the other levels by tracing the delegation of a goal backward. A want-level may usually take the value of 10 because most actors want a goal to be achieved as well as possible. When the expectation of an actor is downgraded, the corresponding want-level may decrease. In Fig. 2, the want-level of G2 is five because the "Submission system" can wait for his/her submission until the deadline.

The second issue is how to decide on the quality characteristics of each goal. For this, the analyst must also make a subjective decision. When a goal is decomposed into other goals, the quality characteristics in a goal are usually inherited into its refined goals. The refined goals are then delegated to other actors. These inheritance and delegation sometimes help the analyst find missing quality characteristics to be attached.

The third issue involves the range of metrics comparability. Metrics are comparable only when the models are about the same activity because they are used to observe the changes of actors and their contribution to the activity. A comparison between the results of the metrics about the models of different activities is meaningless. The values are in ordinal scale; hence, we cannot say that a model is twice or N-times better than another.

The fourth issue is how to decide on the boundary of an activity. The boundary is expected to be suitably decided. However, we did not mind it that much because we focused on the changes of the models about an activity. Even if the model boundary is unsuitable, the changes of the metrics caused by those of the elements in the model tell us whether or not the changes are good.

The fifth issue involves the exploration of an actual goal. People usually do not know their actual goal; they simply complain that their immediate tasks do not work well. Goal-oriented modeling techniques generally contribute to finding an actual goal from these immediate tasks. However, our goal dependency modeling does not focus on this issue. "OriginGoals" should be carefully chosen before our goal dependency models are developed.

The sixth issue is about the difficulty of a following

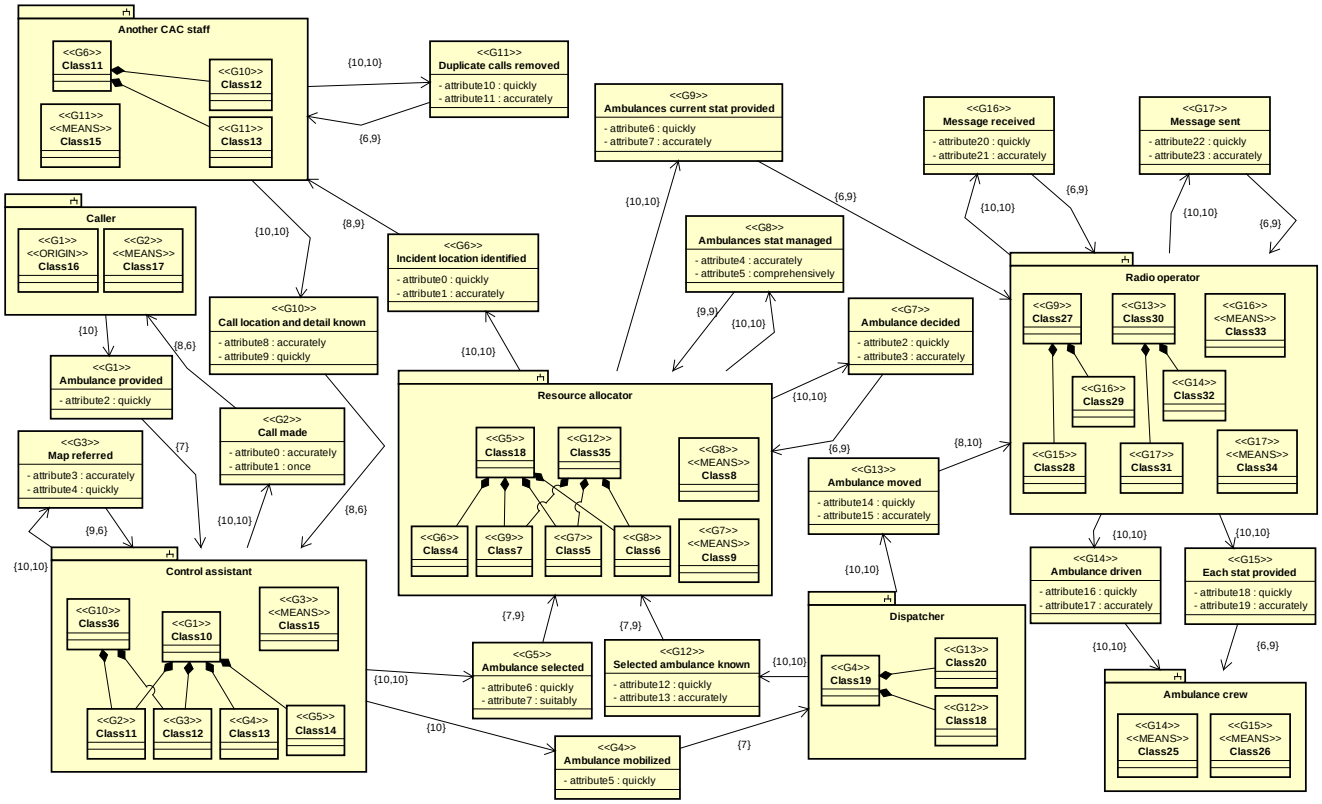


Fig. 11 As-is model of the ambulances dispatch

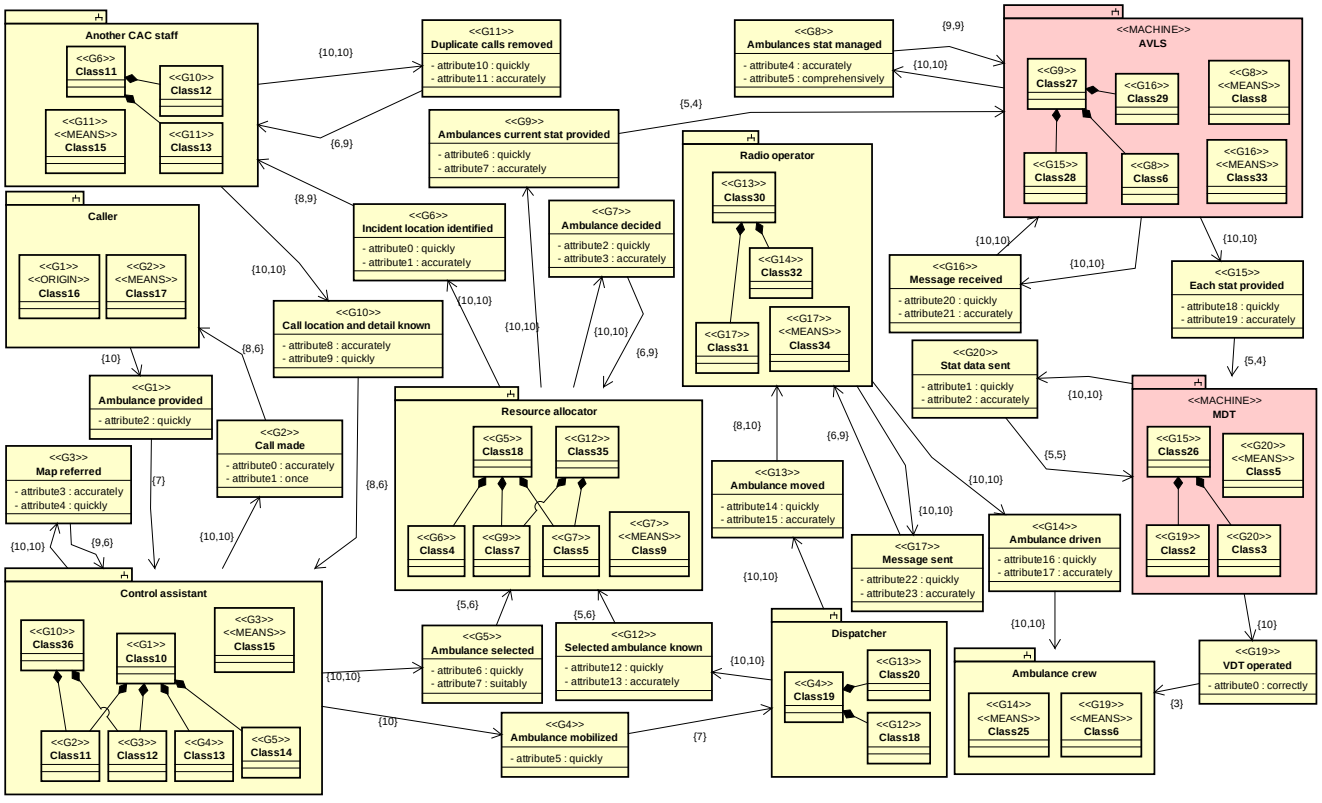


Fig. 12 To-be model of the ambulances dispatch

goal dependency. Even in the small example in Fig. 1, goal G1 is delegated and decomposed, and its sub-goals are finally achieved by three different actors. Such a dependency must be followed because we must confirm that a goal has been achieved by suitable actors. However, people are generally not good at following this dependency. Tool support is crucial in this issue.

The seventh issue touches on the metrics and the or-decomposition inside an actor. Some goal dependencies are not used in operation if the goal hierarchy inside each actor contains or-decompositions. As shown in the formal algorithms in the Appendix, the hierarchy is not considered when the metrics are calculated. The metrics consider all the goal dependencies even if or-decompositions exist. We believe that this decision is valid because the or-decompositions in the model allow the actors in operation to choose any goal decomposition.

The eighth issue is about the effectiveness of the meta-model and constraints in the case study in the previous section. The meta-model and the constraints are the bases for checking the syntax of GDMA models automatically. The syntax checker in section 3 is developed on the basis of the meta-model and the constraints. Therefore, we may regard that the meta-model and the constraints are effective if the cases using the syntax checker are better than the cases without the checker. One of authors performed the cases in the previous section. In [3], other cases were also performed by the same author. Because the author is of course an expert of GDMA, the quality of GDMA models in both cases was almost the same. However, the author could develop the models in the previous section faster than he did in [3]. Although the spending time was not recorded in both cases, the argument about this comparison is obvious because of the following reasons. First, the syntax checker was used in the cases in the previous section, and the checker automatically finds the syntax errors and notifies its users. Second, the author should check the syntax errors manually in cases in [3] by checking the GDMA's documents such as the paper [3] itself written in a natural language. We also introduce the results in an experiment in [4] to show the effects of the meta-model and the constraints to non-experts. If non-experts think the syntax checker is useful, we may regard the meta-model and the constraints are also effective. In the experiment, nine students developed GDMA models, and they were non-experts of GDMA. After their development, we sent out several questionnaires to the student about GDMA including the usability of the syntax checker. Six out of them answered that the checker was useful. We assume few students could develop the models without the syntax checker. Two out of them also answered that they could not understand the error messages of the syntax checker. This problem was not caused by the meta-model and the constraints but by the implementation of the checker. We thus have to improve the understandability of the messages.

The ninth issue is about the advantages of the meta-model and the constraints. We show two advantages as follows. First, anyone can develop case tools of GDMA in any platforms correctly with the meta-model and the constraints. In our previous works [3]–[5], the syntax of GDMA was just explained in a natural language. This could cause misunderstanding about the GDMA models, and mistakes in implementing supporting tools. Second, anyone can correctly understand how the tools in the previous section work with the meta-model and the constraints. Users of the tools could misunderstand or could not correctly understand what the tools do only the informal explanations in [3]–[5]. As we mentioned in the previous paragraph, two out of nine students reported they could not understand the messages of the syntax checker in cases in [4]. We assume that such students could understand them if they had learnt the meta-model and the constraints beforehand.

The last issue is the threats to the validity of our evaluation. Threat to internal validity can affect the independent variables of the case studies. The metrics are derived from the measureable values in each model, such as the number of the specific type of elements and values attached to an association between elements. Therefore, such values are the independent variables. Although the values are decided on the basis on the reports in the literatures, the decision was made by the authors subjectively. Threat to internal validity thus exists. Because only two cases are shown, it is hard to say the results are general enough. Therefore, threat to external validity also exists. However, this threat is mitigated a little bit because two different types of cases are shown: one is a successful case, and another is failure case. Construct validity is the degree to which the variables measure the concepts they are to measure. In this validity, variables correspond to metrics we defined. Because our metrics agree with the discussion in the literatures, threat to construct validity does not exist. Because the number of cases is a few, it is impossible to use statistical test. Threat to conclusion validity thus exists.

6. Related work

We regard that a goal between two actors shows the delegation of the goal achievement from an actor to another.

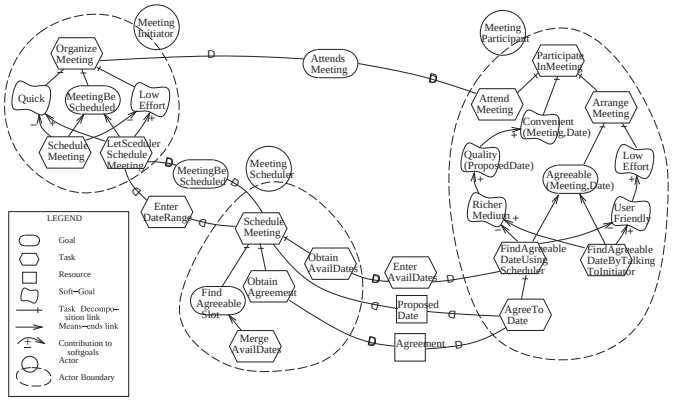


Fig. 13 iStar model in [1]

Therefore, a goal in an actor should be the same as that in another actor when the goal is delegated from the first actor to the second actor. However, the original iStar [1], iStar2.0 [8] and its popular variations [9] (e.g., GRL [10] and Tropos [11]) do not represent a goal dependency in such a way. This is the main reason why we developed our own notation in GDMA. In iStar and its variations, the origin of a goal may be different from the end of the goal for each dependency. Fig. 13 shows a famous example of the iStar model in [1]. In the figure, the "Enter AvailDates" task shows the dependency between an actor "Meeting Scheduler" and another actor "Meeting Participant." However, the dependency's origin of is "Obtain AvailDates," and the end is "Find Agreeable Date Using Scheduler." Both the origin and the end are different from "Enter AvailDates." Note that the goals, tasks, resources, and soft-goals are called intentions, that play a similar role in iStar and its variations. When a human examines the model in Fig. 13, he/she is able to guess the meaning. On the contrary, computer tools cannot guess it. Therefore, we enforce that a goal in an actor is the same as that in another actor when the goal is delegated from the first actor to the second actor, as exemplified in Figs. 1 and 2.

In GDMA, the model of an activity is changed according to the changes of the actors, their expectations, and their abilities (e.g., a change caused by system introduction). Each mode snapshot during the changes is evaluated using the metrics. The changes of the metrics revealed whether the change is good or bad for the people in the activity. Our approach is a quantitative technique, a kind of reasoning approach. According to a systematic literature review [2], there are not so many quantitative approaches (i.e., approximately 15% out of the reviewed papers). Note that we regard a quantitative approach to be contained either "Mathematical Formula" and "Algorithm" category in [2]. GRL [10] encourages quantitative approaches because they have several quantitative elements and enable us to develop some metrics. However, we cannot find metrics similar to those in our research. GRL mainly quantifies the goal satisfaction. Although these results were used to specify the quantitative evaluation of each goal, the whole activity containing the actors was not focused on. We can find other several quantitative approaches. In [12], a quantitative technique was used to select a cloud platform with respect to security and privacy. In this paper, the satisfaction of security and privacy goals were modeled and measured. In [13], a risk analysis was proposed by using goal models. Quantitative values, such as risk impacts, were derived from the structure of a goal model and the values annotating that model. In [14], the KAOS-based goal model of a system was used to assess system implementation and operation. In [15], the requirements were prioritized using a goal model with respect to several stakeholders. The paper used our previous approach [16]. All these studies had different objectives from our research.

Most goal models contain a goal decomposition hierarchy, and some properties of a goal in the hierarchy are propagated according to the hierarchy. Several techniques systematically specify such a propagation mechanism. KAOS [17] used the temporal logic for this mechanism. The approach seems to be the most rigorous approach. GRL enabled us to specify this mechanism using qualitative and quantitative values. The degree of contribution by the sub-goals was quantified in our previous approach [18]. In GDMA, this propagation mechanism was maintained in lightweight manner, as shown in the function quality tracer in Section 3.2.

In many goal-oriented modeling approaches, such as KAOS and iStar and its variations, the conflicts among goals are handled. The conflict handling mechanism enables us to describe a realizable model. We want to add this kind of mechanism in GDMA.

We then reviewed works related to CASE tools. A profile in the UML provided a generic extension mechanism for customizing UML models for particular domains and platforms. UML profiles for goal-oriented modeling are already available. In fact, [19] proposed a UML profile for KAOS. In [20], a UML profile for goal-oriented and use case-driven representation was proposed. In [21], a profile for iStar was presented. Although theoretical merits exist for UML profile usage, most CASE tools do not completely accept these profiles. Therefore, we cannot practically describe models using the profiles. We did not consider UML profiling when we started to develop our own CASE tool.

A survey paper of iStar and its variation tools summarized and compared several tools [22]. In the paper, six famous tools (i.e., OpenOME[23], TAOM4E[24], GR-Tool[25], STS-Tool[26], jUCMNav[27], and DesCARTES[28]) were reviewed with respect to syntactic and semantic viewpoints. According to the review, three were Java programs, and the remaining were Eclipse plugins. Therefore, users must learn how to use them from scratch. Eclipse normally requires very fast machines, and the plugins based on the Eclipse Modeling Framework (EMF) usually require additional machine power. Even if a user knows how to use Eclipse, the user cannot easily know how to use unfamiliar graphical editors based on the EMF. In contrast, our CASE tool can be easily introduced if a user already uses an UML editor, which is the basis of our CASE tools. The survey paper indicated that the syntax coverage of some tools was not good. Our tool completely covered our own syntax rules. The semantic coverage of some tools was also not good. By contrast, our tool completely covered our own semantics because it has to calculate the metrics. Another reason for the good coverage of our tool is the simplicity of our notation.

We summarize the comparison among modeling notations including GDMA in Table 5 with respect to the following criteria.

1. A goal in depender, a depended goal and a goal in dependee are the same.
2. Quality is the first class element.
3. Elements in a model are compared quantitatively.
4. Models are compared quantitatively.
5. Goal decomposition is guided.

Table 5 Comparison among notations (*1: It depends on each variation. *2: KAOS does not have goal dependency.)

Notation/criteria	1	2	3	4	5	6	7
GDMA	Y	N	Y	Y	Y	N	Y
Original Istar	N	Y	N	N	N	Y	N
iStar 2.0	N	Y	N	N	N	Y	N
GRL	N	Y	Y	N	Y	Y	N
Tropos	N	Y	N	N	N	Y	N
Typical IStar variations	N	Y	N/A*1	N/A*1	N/A*1	Y	N
KAOS	N/A*2	Y	N	N	Y	Y	N

6. Conflicted goals are managed.
7. Tools are familiar to usual software engineers.

The term and the concept of the machine in GDMA are imported from the problem frames [29]. A goal in GDMA is specified on the basis of the behavioral goal types in KAOS [17]. A quality attribute corresponds to the soft goal in KAOS.

7. Conclusion

The effects of introduced systems should be carefully examined at the early stage of requirements analysis. In this study, we proposed and evaluated a modeling notation and its analysis, called GDMA. We focused on a goal and two actors who want to achieve the goal and who can achieve the goal. An actor can either be a human, an organization, or an artificial element, such as an information system. The dependencies of the two actors about achieving a goal are represented in a model. We can observe a change of actors, their expectations, and abilities by using the model metrics. We can then decide whether the change is good or bad both quantitatively and qualitatively for the people. To evaluate GDMA, we described the models of system introduction for two cases reported in the literatures and explained the effects caused by the introduction for each case. CASE tools are crucial in efficiently and accurately performing the analysis using the notation. Therefore, we developed tools by extending an existing UML modeling tool.

Acknowledgments

This work was supported by JSPS KAKENHI Grant Numbers 21K11837 and 18K11249.

References

[1] E.S.K. Yu, "Towards modeling and reasoning support for early-phase requirements engineering," 3rd IEEE International Symposium on Requirements Engineering (RE'97), January 5-8, 1997, Annapolis, MD, USA, pp.226–235, IEEE Computer Society, 1997.

[2] E.J.T. Gonçalves, J. Castro, J. Araújo, and T. Heineck, "A systematic literature review of istar extensions," J. Syst. Softw., vol.137, pp.1–33, 2018.

[3] H. Kaiya, S. Ogata, S. Hayashi, and M. Saeki, "Early requirements analysis for a socio-technical system based on goal dependencies," New Trends in Software Methodologies, Tools and Techniques - Proceedings of the Fifteenth SoMeT_16, Larnaca, Cyprus, 12-14 September 2016, ed. H. Fujita and G.A. Papadopoulos, Frontiers in Artificial Intelligence and Applications, vol.286, pp.125–138, IOS Press, 2016.

[4] H. Kaiya and K. Haga, "A CASE tool for goal dependency model with attributes based on an existing UML editor," Knowledge-Based and Intelligent Information & Engineering Systems: Proceedings of the 21st International Conference KES-2017, Marseille, France, 6-8 September 2017, ed. C. Zanni-Merk, C.S. Frydman, C. Toro, Y. Hicks, R.J. Howlett, and L.C. Jain, Procedia Computer Science, vol.112, pp.1196–1205, Elsevier, 2017.

[5] H. Kaiya, W. Fujita, R. Yamada, A. Hazeyama, S. Ogata, T. Okubo, N. Yoshioka, and H. Washizaki, "Experimental evaluation of traceability checking tool for goal dependency modeling," Knowledge-Based Software Engineering: 2020, Proceedings of the 13th International Joint Conference on Knowledge-Based Software Engineering (JCKBSE 2020), Larnaca, Cyprus, August 24-26, 2020, ed. M. Virvou, H. Nakagawa, and L.C. Jain, Learning and Analytics in Intelligent Systems, vol.19, pp.70–83, Springer, 2020.

[6] D.R.A. Schallmo and C.A. Williams, Digital Transformation Now! Guiding the Successful Digitalization of Your Business Model, Springer, 2018.

[7] A. Finkelstein and J. Dowell, "A comedy of errors: The london ambulance service case study," IWSSD '96, pp.2–, IEEE Computer Society, 1996.

[8] F. Dalpiaz, X. Franch, and J. Horkoff, "istar 2.0 language guide," CoRR, vol.abs/1605.07767, 2016.

[9] E. Yu, P. Giorgini, N. Maiden, and J. Mylopoulos, Social Modeling for Requirements Engineering, MIT Press, 2010.

[10] D. Amyot, S. Ghanavati, J. Horkoff, G. Mussbacher, L. Peyton, and E.S.K. Yu, "Evaluating goal models within the goal-oriented requirement language," Int. J. Intell. Syst., vol.25, no.8, pp.841–877, 2010.

[11] P. Bresciani, A. Perini, P. Giorgini, F. Giunchiglia, and J. Mylopoulos, "Tropos: An agent-oriented software development methodology," Auton. Agents Multi Agent Syst., vol.8, no.3, pp.203–236, 2004.

[12] H. Mouratidis, S. Islam, C. Kalloniatis, and S. Gritzalis, "A framework to support selection of cloud providers based on security and privacy requirements," J. Syst. Softw., vol.86, no.9, pp.2276–2293, 2013.

[13] F.B. Aydemir, P. Giorgini, and J. Mylopoulos, "Multi-objective risk analysis with goal models," Tenth IEEE International Conference on Research Challenges in Information Science, RCIS 2016, Grenoble, France, June 1-3, 2016, pp.1–10, IEEE, 2016.

[14] C. Ponsard and R. Darimont, "Quantitative assessment of goal models within and beyond the requirements engineering tool: A case study in the accessibility domain," Proceedings of the 10th International i* Workshop co-located with the 29th International Conference on Advanced Information Systems Engineering (CAiSE 2017), Essen, Germany, June 12-13, 2017, ed. S. Ghanavati, L. Liu, and L. López, CEUR Workshop Proceedings, vol.1829, pp.13–18, CEUR-WS.org, 2017.

[15] M. Lencastre and J. Pimentel, "A metamodel for istar-p: Requirements prioritization with goal models," Proceedings of the 12th International i* Workshop co-located with 38th International Conference on Conceptual Modeling (ER 2019), Salvador, Brazil, November 4th, 2019, ed. J. Pimentel, J.P. Carvallo, and L. López, CEUR Workshop Proceedings, vol.2490, CEUR-WS.org, 2019.

[16] H. Kaiya, H. Horai, and M. Saeki, "AGORA: attributed goal-oriented requirements analysis method," 10th Anniversary IEEE Joint International Conference on Requirements Engineering (RE 2002), 9-13 September 2002, Essen, Germany, pp.13–22, IEEE Computer Society, 2002.

[17] A. van Lamsweerde, Requirements Engineering: From System Goals to UML Models to Software Specifications, Wiley, 2009.

[18] S. Hayashi, D. Tanabe, H. Kaiya, and M. Saeki, "Impact analysis on an attributed goal graph," IEICE Trans. Inf. Syst., vol.95-D, no.4, pp.1012–1020, 2012.

[19] W. Heaven and A. Finkelstein, "UML profile to support requirements engineering with KAOS," IEE Proceedings - Software, vol.151, no.1, pp.10–28, 2004.
[20] S. Supakkul and L. Chung, "A UML profile for goal-oriented and use case-driven representation of nfrs and frs," Third ACIS International Conference on Software Engineering, Research, Management and Applications (SERA 2005), 11-13 August 2005, Mt. Pleasant, MI, USA, pp.112–121, 2005.
[21] D. Amyot, J. Horkoff, D. Gross, and G. Mussbacher, "A lightweight GRL profile for i* modeling," Advances in Conceptual Modeling - Challenging Perspectives, ER 2009 Workshops CoMoL, EThe-CoM, FP-UML, MOST-ONISW, QoIS, RIGiM, SeCoGIS, Gramado, Brazil, November 9-12, 2009. Proceedings, pp.254–264, 2009.
[22] C. Almeida, M. Goulão, and J. Araújo, "A systematic comparison of i* modelling tools based on syntactic and well-formedness rules," Proceedings of the 6th International *i** Workshop 2013, Valencia, Spain, June 17-18, 2013, pp.43–48, 2013.
[23] J. Horkoff, Y. Yu, and E.S.K. Yu, "Openome: An open-source goal and agent-oriented model drawing and analysis tool," Proceedings of the 5th International *i** Workshop 2011, Trento, Italy, August 28-29, 2011, pp.154–156, 2011.
[24] M. Morandini, D.C. Nguyen, L. Penserini, A. Perini, and A. Susi, "Tropos modeling, code generation and testing with the taom4e tool," Proceedings of the 5th International *i** Workshop 2011, Trento, Italy, August 28-29, 2011, pp.172–174, 2011.
[25] R. Sebastiani, P. Giorgini, and J. Mylopoulos, "Simple and minimum-cost satisfiability for goal models," Advanced Information Systems Engineering, 16th International Conference, CAiSE 2004, Riga, Latvia, June 7-11, 2004, Proceedings, pp.20–35, 2004.
[26] E. Paja, F. Dalpiaz, M. Poggianella, P. Roberti, and P. Giorgini, "Sts-tool: Specifying and reasoning over socio-technical security requirements," Proceedings of the 6th International *i** Workshop 2013, Valencia, Spain, June 17-18, 2013, pp.131–133, 2013.
[27] D. Amyot, G. Mussbacher, S. Ghanavati, and J. Kealey, "GRL modeling and analysis with jucmnav," Proceedings of the 5th International *i** Workshop 2011, Trento, Italy, August 28-29, 2011, pp.160–162, 2011.
[28] Y. Wautelet and M. Kolp, "Mapping i* within UML for business modeling," Requirements Engineering: Foundation for Software Quality - 19th International Working Conference, REFSQ 2013, Essen, Germany, April 8-11, 2013. Proceedings, pp.237–252, 2013.
[29] M. Jackson, Problem Frames, Analyzing and structuring software development problems, Addison-Wesley, 2000.

Appendix A: Algorithms of metrics

Algorithm 1 Calculate the ANW

Require: Meta-model in Fig. 3, Eqs. (1) to (4)
$hset$ = Set of all instances of Human in a model
$numer = 0$
$denom = hset.size()$
for (Human $h : hset$) **do**
  for (WantGoal $wg : h.wantgoal$) **do**
    if $wg\ instanceof$ OriginGoal **then**
      $numer++$
    end if
  end for
end for
ANW = $numer/denom$

Algorithm 2 Calculate the AGW

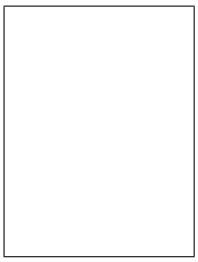
Require: Meta-model in Fig. 3, Eqs. (1) to (4)
$hset$ = Set of all instances of Human in a model
$numer = 0$
$denom = 0$
for (Human $h : hset$) **do**
  for (Want $w : h.want$) **do**
    $ws = w.level$
    $cs = w.dependgoal.can.level$
    $denom += ws.size()$
    $i = 0$
    while $i < ws.size()$ **do**
      $numer += cs[i]/ws[i]$
      $i++$
    end while
  end for
end for
AGW = $numer/denom$

Algorithm 3 Calculate the ANC

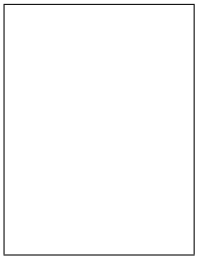
Require: Meta-model in Fig. 3, Eqs. (1) to (4)
$hset$ = Set of all instances of Human in a model
$numer = 0$
$denom = hset.size()$
for (Human $h : hset$) **do**
  for (CanGoal $wg : h.cangoal$) **do**
    if $wg\ instanceof$ MeansGoal **then**
      $numer++$
    end if
  end for
end for
ANC = $numer/denom$

Algorithm 4 Calculate the AGC

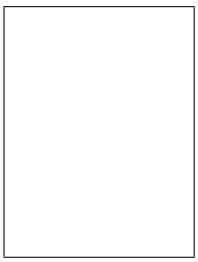
Require: Meta-model in Fig. 3, Eqs. (1) to (4)
$hset$ = Set of all instances of Human in a model
$numer = 0$
$denom = 0$
for (Human $h : hset$) **do**
  for (Can $c : h.can$) **do**
    $cs = c.level$
    $ws = c.dependgoal.want.level$
    $denom += cs.size()$
    $i = 0$
    while $i < cs.size()$ **do**
      $numer += ws[i]/cs[i]$
      $i++$
    end while
  end for
end for
AGC = $numer/denom$



Haruhiko Kaiya is a professor at Kanagawa University, Yokohama, Japan.



Shinpei Ogata is an Associate Professor at Shinshu University, Japan. He received his BE, ME, and PhD from Shibaura Institute of Technology in 2007, 2009, and 2012 respectively. From 2012 to 2020, he was an Assistant Professor, and since 2020, he has been an Associate Professor at Shinshu University. He is a member of IEEE, ACM, IEICE, IPSJ, and JSSST. His current research interests include model-driven engineering for software development.



Shinpei Hayashi is an associate professor in School of Computing at Tokyo Institute of Technology. He received a B.E. degree in information engineering from Hokkaido University in 2004. He also respectively received M.E. and D.E. degrees in computer science from Tokyo Institute of Technology in 2006 and 2008. His research interests include software evolution and software development environment.